Oracle Size: a New Measure of Difficulty for Communication Tasks

Pierre Fraigniaud*
CNRS
Laboratoire de Recherche en Informatique (LRI)
Université Paris-Sud
91405 Orsay, France
pierre@lri.fr

David Ilcinkas*
Laboratoire de Recherche en Informatique (LRI)
Université Paris-Sud
91405 Orsay, France
ilcinkas@lri.fr

Andrzej Pelc†
Département d'informatique
Université du Québec en Outaouais
Gatineau, Québec J8X 3X7, Canada
pelc@uqo.ca

ABSTRACT

We study the problem of the amount of knowledge about a communication network that must be given to its nodes in order to efficiently disseminate information. While previous results about communication in networks used particular partial information available to nodes, such as the knowledge of the neighborhood or the knowledge of the network topology within some radius, our approach is *quantitative*: we investigate the minimum total number of bits of information (minimum oracle size) that has to be available to nodes in order to perform efficient communication.

It turns out that the minimum oracle size for which a distributed task can be accomplished efficiently, can serve as a measure of the difficulty of this task. We use this measure to make a quantitative distinction between the difficulty of two apparently similar fundamental communication primitives: the broadcast and the wakeup. In both of them a distinguished node, called the source, has a message, which has to be transmitted to all other nodes of the network. In the wakeup, only nodes that already got the source message (i.e., are awake) can send messages to their neighbors, thus waking them up. In the broadcast, all nodes can send control messages even before getting the source message, thus potentially facilitating its future dissemination. In both cases we are interested in accomplishing the communication task with optimal message complexity, i.e., using a number of messages linear in the number of nodes.

We show that the minimum oracle size permitting the *wakeup* with a linear number of messages in a n-node network, is $\Theta(n \log n)$, while the *broadcast* with a linear number of messages can be achieved with an oracle of size $O(n)$. We also show that the latter oracle size is almost optimal: no oracle of size $o(n)$ can permit to broadcast with a linear number of messages. Thus an efficient wakeup requires strictly more information about the network than an efficient broadcast.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Network communications*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*Network problems*

General Terms

Algorithms, Theory

Keywords

Broadcast, Wakeup, Oracle

1. INTRODUCTION

1.1 Background and related work

For many network problems (such as leader election, constructing a minimum spanning tree, exploration, wakeup, broadcast, etc.), the quality of the algorithmic solutions often depends on the amount of knowledge given to nodes of the network. For example, if every node knows the topology of the network within radius ρ of it, it is shown in [2] that $\Theta(\min\{m, n^{1+\Theta(1)/\rho}\})$ is the minimum number of messages of bounded length permitting the wakeup of a network with n nodes and m edges. (In [2] the authors talk about the broadcast but their model does not permit transmissions before receiving the source message, hence it is called wakeup in our terminology). Broadcasting time in radio networks is another subject where information available to nodes significantly influences efficiency. In [10] it is shown that if nodes have complete knowledge of the network then deterministic broadcast can be done in time $O(D + \log^3 n)$, for n-node radio networks with diameter D. (This result has been recently improved to $O(D + \log^2 n)$ in [17]). On the other

*Supported by the projects PairAPair of the ACI Masses de Données, and FRAGILE of the ACI Sécurité Informatique. Additional support from the INRIA project "Grand Large".
†Supported in part by NSERC discovery grant and by the Research Chair in Distributed Computing of the Université du Québec en Outaouais.

PODC'06, July 22-26, 2006, Denver, Colorado, USA.

hand, in [4] a lower bound of $\Omega(n \log D)$ is proved on deterministic broadcast time in radio networks in which nodes know only their own identity. (An almost matching upper bound of $O(n \log^2 D)$ is proved in [7]). Another problem, in which partial information about the network significantly influences the efficiency of solutions, is network exploration by a mobile agent. For instance, it is proved in [3] that, if an upper bound $\hat{n}$ on the number n of nodes of an anonymous digraph is known, then a mobile agent can explore this digraph in time polynomial in $\hat{n}$, using one pebble, while without this knowledge, $\Theta(\log \log n)$ pebbles are necessary and sufficient. On the other hand, in [8] the authors investigate the exploration of various types of graphs when the exploring agent is provided with an unlabeled map of the graph, and show how the cost of exploration changes when no map is available.

In fact, the impact of knowledge concerning the environment is significant in many areas of distributed computing, as witnessed by [9, 19] where hundreds of impossibility results and lower bounds for distributed computing are surveyed, many of them depending on whether or not the nodes are provided with partial knowledge of the topology of the network. Finally, notice that the amount of knowledge has also a strong impact on computing in anonymous networks (cf., e.g., [12], where the impact of knowing the total number of nodes is studied in depth).

1.2 The oracle

A network is modeled as an undirected connected graph whose nodes have distinct labels, and ports at any node v of degree $\deg(v)$ are labeled 0,1,...,$\deg(v) - 1$. One distinguished node of the network is called the *source*. A priori, every node has only information concerning itself: it knows its own label (if any), and it knows whether it is the source or not. All additional knowledge available to the nodes of the network (in particular knowledge concerning the rest of the network), is modeled by an *oracle*. An oracle is a function $\mathcal{O}$ whose arguments are networks, and the value $\mathcal{O}(G)$, for a network $G = (V, E)$, is in turn a function $f : V \rightarrow \{0, 1\}^*$ assigning a binary string to every node v of the network. Intuitively, the oracle looks at the entire labeled network and assigns to every node some information, coded as a string of bits. The *size* of the oracle on a given network G is the sum of the lengths of all the strings it assigns to nodes. Hence this size is a measure of the amount of information about the network, available to its nodes.

Solving a network problem $\mathcal{P}$ using oracle $\mathcal{O}$ consists in designing an algorithm that is *unaware* of the network G at hand but solves the problem $\mathcal{P}$ for it, as long as every node v of the network G is provided with the string of bits $f(v)$, where $f = \mathcal{O}(G)$. Typical distributed network problems that may be solved using an oracle are various communication tasks, such as broadcast, wakeup or gossip (information exchange among nodes), as well as, e.g., the construction of a BFS tree or a minimum spanning tree. The formulation of the problem $\mathcal{P}$ may include a demand on the efficiency of the solution, thus we may be interested in communicating within a prescribed time, or constructing a minimum spanning tree using at most a prescribed number of messages. Given the problem $\mathcal{P}$, we ask what is the minimum size of an oracle for solving it. This minimum oracle size can be considered as a measure of the difficulty of the problem $\mathcal{P}$. The novelty and significance of the use of an oracle to model knowledge about the network is that it enables asking *quantitative* questions about the required knowledge, regardless of what *kind* of knowledge is supplied. This should be contrasted with the traditional approach that assumes availability of particular items of information, such as the neighborhood of a node.

It turns out that the minimum oracle size, for which a distributed task can be accomplished efficiently, can be used to make a quantitative distinction between the difficulty of apparently similar problems. We show this for two fundamental communication primitives performing information dissemination: the broadcast and the wakeup from a single source. In both of them a distinguished node, called the source, has a message which has to be transmitted to all other nodes of the network. Nodes send messages along edges of the network. In the wakeup, only nodes that already got the source message (i.e., are awake) can send messages to their neighbors, thus waking them up. In the broadcast, all nodes can send control messages even before getting the source message, thus potentially facilitating its future dissemination. In both cases we are interested in accomplishing the communication task with optimal message complexity, i.e., using a number of messages linear in the number of nodes. We ask what is the minimum size of an oracle permitting to do that.

An approach similar to ours has been developed in the context of *informative labelings*. Informative labeling schemes are ways to label the nodes of a network with short labels in such a way that queries such as inter-node distance [11], ancestor [1], connectivity [13], etc., can be answered based solely on the labels of the nodes involved in the query. The main objective in this context is to design schemes using short labels, and guaranteeing that queries can be rapidly answered. The oracle terminology is sometimes also used in the context of informative labeling (e.g., when the query can be answered in constant time [20]). Conversely, the informative labeling terminology is sometimes also used for problems involving global properties [16].

Our point of view is to use the informative labeling terminology in the context of distributed data-structures enabling quick answers to queries, and to use the oracle terminology in the context of distributed computing when nodes have to collaborate to achieve complex tasks (e.g., broadcasting, coloring, wake-up, leader election, etc.). One reason motivating this view is, for instance, that giving the knowledge of the network size n to the nodes can hardly be seen as labeling every node by n. Also, the notion of oracle easily extends to the case when the information is not given a priori, but on line, during the execution of the protocol.

1.3 Our results

We show that the minimum oracle size permitting the *wakeup* with a linear number of messages in a network with at most n nodes, is $\Theta(n \log n)$, while the *broadcast* with a linear number of messages can be achieved with an oracle of size $O(n)$. We also show that the latter oracle size is almost optimal: no oracle of size $o(n)$ can permit to broadcast with a linear number of messages. Thus an efficient wakeup requires strictly more information about the network than an efficient broadcast.

Our upper bounds are constructive: we show specific oracles of appropriate size and design wakeup and broadcast algorithms using them and accomplishing information dissemination with a linear number of messages. Apart from their tightness, our results have the following additional strength. Both upper bounds hold even for totally asynchronous communication, for anonymous nodes (no distinct labels), and using only bounded-size messages. On the other hand, both lower bounds hold even for synchronous communication, for labels of nodes 1,...,n, and for arbitrarily long messages.

We consider wakeup from a single source, where only one node, the source, is awake in the beginning. We choose this communication primitive due to its similarity with broadcasting, since we want to compare the amount of information needed to efficiently accomplish similar tasks. However, both the upper and the lower bound on oracle size for wakeup still hold for a more traditional formulation of the wakeup problem, where the adversary wakes up an arbitrary subset of nodes which in turn have to wake up all other nodes.

1.4 Terminology and preliminaries

We now describe broadcast algorithms using oracles, in a more detailed manner. Wakeup algorithms will be a particular type of broadcast algorithms, subject to an additional constraint. Consider a network, i.e., a connected graph $G = (V, E)$ with a distinguished source s. Every node v of degree $\deg(v)$ has a distinct label $\mathrm{id}(v)$, and ports at v are labeled $0, 1, ..., \deg(v)-1$. The port at node v, corresponding to edge e, is denoted by $\mathrm{port}_v(e)$. Every node v has also a bit $s(v)$ called the *status bit*, which is set to 1 if the node v is the source, and to 0 otherwise. Fix an oracle $\mathcal{O}$ and let $f = \mathcal{O}(G)$ be a function $f : V \to \{0,1\}^*$, assigning binary strings to nodes of G. A broadcast algorithm $\mathcal{A}$ using oracle $\mathcal{O}$ is a function $\mathcal{A} : \{0,1\}^* \times \{0,1\} \times \mathbb{N} \times \mathbb{N} \to \Sigma$, where $\mathbb{N}$ denotes the set of non-negative integers and Σ denotes the set of *broadcast schemes* (to be defined below). For a given node v, algorithm $\mathcal{A}$ takes the quadruple $(f(v), s(v), \mathrm{id}(v), \deg(v))$ and returns a broadcast scheme $S_v = \mathcal{A}(f(v), s(v), \mathrm{id}(v), \deg(v))$ for the node v. It remains to define what such a scheme is. Intuitively, this is a prescription whether and on which ports the node should send messages, and what messages, given a particular history of communication to date. Such a history (at node v) is a sequence $H = (f(v), s(v), \mathrm{id}(v), \deg(v), (m_1, p_1), (m_2, p_2), \ldots, (m_k, p_k))$, where the prefix $(f(v), s(v), \mathrm{id}(v), \deg(v))$ is the knowledge of the node before the broadcast starts, and $(m_1, m_2, \ldots, m_k)$ are messages already received by v, where m_i came to node v on port p_i. Intuitively, the history describes the total knowledge of the node at a given point of the broadcast process. Given a history H at node v, a broadcast scheme S_v returns a set of couples $\{(m'_1, p'_1), \ldots (m'_r, p'_r)\}$, where $0 \le p'_i < \deg(v)$. This means that v should send message m'_i on port p'_i, for $i \le r$. At each point of the scheme execution some nodes are *informed*. Intuitively, these are nodes that already got the source message. In the beginning only the source is informed. A node becomes informed after receiving a message from an informed node (indeed, the source message can be appended to any such message). Broadcast is completed when all nodes of the network are informed. Wakeup algorithms using oracles produce wakeup schemes in a similar manner as above: a wakeup scheme for v is a broadcast scheme that does not send any messages (returns the empty set) on all histories with no messages, unless v is the source. Intuitively, nodes other than the source can spontaneously transmit in the broadcast but they cannot in the wakeup. The message complexity of a broadcast or a wakeup scheme is the total number of messages that it produces.

2. ORACLE SIZE FOR THE WAKEUP

In this section we show that the minimum oracle size permitting the wakeup with a linear number of messages is $\Theta(n \log n)$. Establishing the upper bound is easy. Fix a network G, and let T be any spanning tree of G. The value f of oracle $\mathcal{O}$ on the network G is defined as follows. For any node v, $f(v)$ is a binary string coding those port numbers at v that lead to its neighbors in T. Since port numbers are integers smaller than n, there exists such a string of length at most $n(v)\lceil \log n \rceil + O(\log \log n)$, where $n(v)$ is the number of neighbors of v in T. (One way of constructing such an encoding string is this. Concatenate all binary representations of the $n(v)$ port numbers that lead to the neighbors of v in T, using exactly $\lceil \log n \rceil$ bits for each of them. Let the obtained sequence α have length $x = n(v)\lceil \log n \rceil$. Construct another binary sequence β as follows. Let $b_1 \ldots b_r$ be the binary representation of $\lceil \log n \rceil$. Let $\beta = b_1 b_1 b_2 b_2 \ldots b_r b_r 10$. Finally, concatenate α to β. The obtained sequence has length $x + O(\log \log n)$, and the original representations of the $n(v)$ port numbers are easy to obtain from it.) Hence the size of the oracle is $n \log n + o(n \log n)$. Given this oracle, the wakeup scheme at v tells the node to send messages on all ports coded by $f(v)$. This scheme uses exactly $2(n-1)$ messages. Thus we have:

THEOREM 1. *There exists an oracle of size $O(n \log n)$ permitting the wakeup with a linear number of messages of networks with at most n nodes.*

The main result of this section establishes a matching lower bound on oracle size for this task.

THEOREM 2. *The minimum oracle size permitting the wakeup with a linear number of messages of networks with at most n nodes must have size $\Omega(n \log n)$.*

To prove this theorem, we use an auxiliary problem, called `edge_discovery`, defined as follows. We denote by K_n^* the n-node complete graph K_n whose vertices are labeled from 1 to n, and the port number at node i of the edge leading to node j is labeled $(i-j) \bmod (n-1)$. The instances of the problem `edge_discovery` are triples (n, X, Y) where n is a positive integer, and X and Y are two disjoint subsets of edges of K_n^*. In fact, the edges in X are given distinct labels between 1 and $|X|$. So formally, $X = \{(e_1, \ell_1), \ldots, (e_{|X|}, \ell_{|X|})\}$, where ℓ_i is the label of e_i. The problem consists in designing a communication scheme that, given n, $|X|$, and Y, eventually discovers X. Whenever an edge e is traversed by a message of the communication scheme, the following information is obtained: if $(e, \ell) \in X$ then (e, ℓ) is revealed; otherwise it is revealed that $(e, \ell) \notin X$ for any label ℓ.

We prove the following lemma that will be used later as a key tool for proving our lower bounds. In the absence of any information about the instances, this lemma could be proved using techniques similar to those in, e.g., [14] and [15]. However, the presence of an oracle changes the setting radically.

LEMMA 1. *Let $\mathcal{I}$ be a subset of instances of* `edge_discovery`*, all yielding the same input for the problem (i.e., these instances differ only in the sets X, and all these sets X*

have the same size). The worst-case message complexity of `edge_discovery` *restricted to* $\mathcal{I}$ *is at least* $\log \frac{|\mathcal{I}|}{|X|!}$.

PROOF. For any given instance (n, X, Y) of `edge_discovery`, the edges in X are called *special*. Let us consider any communication scheme S solving `edge_discovery` for all instances in $\mathcal{I}$. Messages traverse edges during the execution of S. At the beginning of the execution of the scheme, all instances in $\mathcal{I}$ are called *active*. When an edge is traversed, the scheme learns whether this edge is special or not. This knowledge enables to discard instances from the set of active instances. For example, if the traversed edge e is not special, then all currently active instances in which e is special can be discarded. Conversely, if the traversed edge e is special, then all currently active instances in which e is not special can be discarded.

We describe an adversary that aims at slowing down the discovery of the special edges by S. We consider the synchronous execution of S. A set $J \subseteq \mathcal{I}$ of active instances, after t messages have been sent by S and r special edges have been discovered by S, for some $t \geq 0$ and $r \geq 0$, is said to be *uniform* if (1) the t first messages are sent by S through the same edges in all instances in J, and (2) the set of revealed couples (e, ℓ), for special edges e, is the same in all instances of J. If J is uniform, then the scheme S will proceed identically at the next step of its execution in all instances of J. That is, a message is sent through edge e and this edge is the same for all instances in J. The adversary considers uniform sets of active instances, and proceeds as follows. Let $J_{special}$ (resp., $J_{regular}$) be the set of instances in a uniform set J, for which e is special (resp., not special). If $|J_{special}| \geq |J_{regular}|$ then the adversary decides that e is special, else it decides that e is not special. Note that $|J_{special}| \geq \frac{1}{2}|J|$ in the former case, and $|J_{regular}| \geq \frac{1}{2}|J|$ in the latter case. In case e is set to be special by the adversary, the label $\ell(e)$ remains to be set. The adversary proceeds as follows. Since r special edges have been already discovered, the label of e can take $|X| - r$ values. The adversary chooses the label l_0 such that the set $J_{special}^{(l_0)}$ of active instances in $J_{special}$ for which $\ell(e) = l_0$, has the largest size. Note that then $|J_{special}^{(l_0)}| \geq \frac{|J|}{2(|X|-r)}$. We say that $J_{regular}$ is the *regular* subset of J, and $J_{special}^{(l_0)}$ is the *special* subset of J.

By construction, $J_{regular}$, as well as all $J_{special}^{(l)}$ for $1 \leq l \leq |X|$, are uniform. Hence, we can define recursively the following sequence of sets: $I_{0,0} = \mathcal{I}$ and

$$\begin{cases} I_{t+1,r} = \text{the regular subset of } I_{t,r}, \\ \qquad\qquad \text{if the } (t+1)\text{th edge is set as not special;} \\ I_{t+1,r+1} = \text{the special subset of } I_{t,r}, \\ \qquad\qquad \text{if the } (t+1)\text{th edge is set as special.} \end{cases}$$

By construction, depending on which of the cases holds, we have $|I_{t+1,r}| \geq |I_{t,r}|/2$ or $|I_{t+1,r+1}| \geq |I_{t,r}|/(2(|X|-r))$. For the above defined adversary, let $x_{t,r}$ denote the number of active instances after t messages have been sent in S, and r special edges have been discovered. Thus, $x_{0,0} = |\mathcal{I}|$ and

$$x_{t+1,r} \geq \frac{x_{t,r}}{2} \ \text{ and } \ x_{t+1,r+1} \geq \frac{x_{t,r}}{2(|X|-r)}\,.$$

Therefore, by simple induction on r and t, we get

$$x_{t,r} \geq \frac{x_{0,0}\ (|X|-r)!}{2^t\ |X|!}\,.$$

As a consequence, $x_{t,r} \geq x_{0,0}/(2^t|X|!)$ for any $r \leq |X|$. When the communication scheme S is completed, only one instance remains active, i.e., $x_{t,r} \leq 1$. By the previous inequality, our adversarial scenario guarantees that this cannot occur before t messages have been sent, where

$$\frac{x_{0,0}}{(2^t|X|!)} \leq 1\,.$$

Since $x_{0,0} = |\mathcal{I}|$, we conclude that $t \geq \log \frac{|\mathcal{I}|}{|X|!}$, which completes the proof of Lemma 1. $\square$

PROOF OF THEOREM 2. We first prove that, for any $\alpha < 1/2$, there exists $\epsilon > 0$ such that, for any integer n greater than some constant, there exists a $(2n)$-node graph for which no algorithm can perform wakeup with less than $\epsilon\,(2n)\log(2n)$ messages, if the oracle size is not more than $\alpha\,(2n)\log(2n)$.

Fix a positive integer n. Recall that K_n^* is the n-node complete graph with the following labeling. The nodes of K_n^* are labeled from 1 to n. The port numbers of the edges are fixed as follows: for any $1 \leq i, j \leq n$, the port number at i of the edge $\{i, j\}$ is $(i - j) \bmod (n - 1)$.

For any n-tuple $S = (e_1, e_2, \ldots, e_n)$ of distinct edges in K_n^*, let $G_{n,S}$ be the graph defined from K_n^* as follows. For any $1 \leq i \leq n$, a node w_i labeled $n + i$ is inserted in the middle of the edge $e_i = \{u_i, v_i\}$. The port number at u_i (resp. at v_i), of the edge $\{u_i, w_i\}$ (resp. $\{v_i, w_i\}$), is the same as the port number at u_i (resp. at v_i), of the former edge $\{u_i, v_i\}$. Assume, without loss of generality, that the label of u_i is smaller than the label of v_i. Then the port number at w_i of the edge $\{u_i, w_i\}$, (resp. $\{v_i, w_i\}$), is 0 (resp. 1). Other port numbers remain unchanged. Let node with label 1 be the source.

Intuitively, an oracle has to give a lot of information to help a wakeup algorithm to find the n subdivided edges with only $O(n)$ messages. This is mainly due to the fact that there exists a lot of different graphs $G_{n,S}$. The graphs $G_{n,S}$ are indeed distinct for different sets S. There are $P = n!\binom{\binom{n}{2}}{n}$ such (labeled) graphs, as there are $\binom{n}{2}$ edges in K_n^*. Let us compute a lower bound on P. First note that, for any a, b such that $1 \leq b \leq a$, we have

$$\binom{a}{b} \geq \left(\frac{a}{b}\right)^b\,. \tag{1}$$

This implies

$$\binom{\binom{n}{2}}{n} \geq \left(\frac{\binom{n}{2}}{n}\right)^n\,.$$

Moreover, we have $\binom{n}{2} = \frac{n(n+1)}{2} \geq n^2/2$. Hence

$$P \geq n!\left(\frac{n}{2}\right)^n\,. \tag{2}$$

Consider an arbitrary wakeup algorithm using an oracle $\mathcal{O}$. Assume that $\mathcal{O}$ has size at most $q = \alpha(2n)\log(2n)$ on all graphs of size $2n$, for some $\alpha < 1/2$. We will prove that there are many graphs $G_{n,S}$ for which the oracle has the same output.

Let us first compute how many different functions f an oracle $\mathcal{O}$ of size at most q can output for $(2n)$-node graphs. Let $v_1, \ldots, v_{2n}$ be the list of nodes of such a graph, in increasing order of their identifiers. Consider a function $f = \mathcal{O}(G)$. For any $1 \leq i \leq 2n$, $f(v_i)$ is the (possibly empty) string given by the oracle to the node v_i in the graph G. Let s be the concatenation of the strings $f(v_i)$ in increasing order of i. Let q' be the size of s. By definition of q, we have $q' \leq q$. There are $2^{q'}$ possible different strings s for a given q'. Moreover, using standard combinatorial arguments, one can show that there are $\binom{q'+2n-1}{2n-1}$ different ways to partition the q' bits of the string s into $2n$ (possibly empty) substrings $f(v_i)$. To summarize, an oracle of size q' can output $2^{q'}\binom{q'+2n-1}{2n-1}$ different functions for $(2n)$-node graphs. Since q' can be chosen between 0 and q, the oracle has exactly

$$Q = \sum_{q'=0}^{q} \left(2^{q'} \binom{q'+2n-1}{2n-1} \right)$$

possible different outputs.

Let us compute an upper bound on Q. Since $2^{q'}\binom{q'+2n-1}{2n-1}$ is increasing as a function of q', it follows that Q is at most $(q+1)\left(2^q\binom{q+2n-1}{2n-1}\right)$. Note that $\binom{q+2n-1}{2n-1} \leq \binom{q+2n}{2n}$ because $q \geq 0$. Thus we have

$$Q \leq (q+1)\left(2^q \binom{q+2n}{2n}\right) \tag{3}$$

Recall that $q = \alpha(2n)\log(2n)$. Thus we have $\binom{q+2n}{2n} = \binom{2n(1+\alpha\log(2n))}{2n}$.

In view of Lemma 1.6 of [18] stating that

$$\binom{a}{b} \leq \left(\frac{ae}{b}\right)^b \tag{4}$$

for $0 < b \leq a$, we get

$$\binom{2n(1+\alpha\log(2n))}{2n} \leq \Big(e\big(1+\alpha\log(2n)\big)\Big)^{2n}.$$

For n large enough, we have

$$\Big(e\big(1+\alpha\log(2n)\big)\Big)^{2n} \leq \big(6\alpha\log(2n)\big)^{2n}$$

and thus

$$Q \leq \big(\alpha\,(2n)\log(2n)+1\big)\cdot 2^{\alpha\,(2n)\log(2n)}\cdot\big(6\alpha\log(2n)\big)^{2n}.$$

Take $\beta = 1/4 + \alpha/2$. We have $\alpha < \beta$, and thus for n large enough,

$$Q \leq 2^{2\beta n\log(n/2)}. \tag{5}$$

The oracle can output at most Q different functions for the P different graphs $G_{n,S}$. Therefore, there exists a function f such that the oracle will output f for a set $\mathcal{G}$ of at least P/Q different graphs $G_{n,S}$. For all these graphs, the wakeup scheme returned by the algorithm is the same.

To any graph $G_{n,S} \in \mathcal{G}$ we can associate an instance $(n, S, \varnothing)$ of the `edge_discovery` problem, where the special edges of `edge_discovery` are the subdivided edges of the graph $G_{n,S}$, and the label of a special edge is the rank of the subdivided edge in S. Let $\mathcal{I}$ be the set of instances of `edge_discovery` obtained from all graphs in $\mathcal{G}$. Clearly, $\mathcal{I}$ and $\mathcal{G}$ have the same cardinality. Performing wakeup in a graph $G_{n,S}$ requires that, for any $e \in S$, at least one message be sent to the node hidden in the edge e. Moreover, this node has an identifier that corresponds to the position of e in S. Therefore, performing wakeup in a graph $G_{n,S}$ requires at least the same number of messages as solving the `edge_discovery` problem on the corresponding instance.

Combining Equations 2 and 5, we get

$$P/Q \geq n!\; 2^{(1-2\beta)n\log(n/2)}.$$

Since $|\mathcal{I}| \geq P/Q$, the application of Lemma 1 gives a worst-case message complexity of at least

$$\log\left(\frac{n!\; 2^{(1-2\beta)n\log(n/2)}}{n!}\right) = (1-2\beta)\,n\log(n/2) \tag{6}$$

Since $\alpha < 1/2$, we have $\beta < 1/2$ and thus the above message complexity is greater than $\epsilon\,(2n)\log(2n)$ for n large enough, where ϵ is a positive constant not depending on n.

We can now conclude the proof of the theorem. Assume that the theorem does not hold. Then there exists an infinite increasing sequence of integers $(n_i)_{i\geq 1}$, an oracle of size less than $\frac{1}{4}n_i\log n_i$ for the graphs with at most n_i nodes, $i \geq 1$, and an algorithm $\mathcal{A}$ using this oracle, such that the algorithm performs wakeup with a linear number of messages in any graph. Fix $i \geq 1$. Let $m_i = n_i$ if n_i is even and $m_i = n_i - 1$ otherwise. For graphs of size at most m_i, the oracle has size at most $\frac{1}{4}n_i\log n_i$. For i large enough, we have $\frac{1}{4}n_i\log n_i \leq \frac{1}{3}m_i\log m_i$. Applying the previous result with $\alpha = 1/3$, there exists a positive constant ϵ such that $\mathcal{A}$ has a worst-case message complexity of at least $\epsilon\, m_i\log m_i$ on m_i-node graphs, for i large enough. Thus the message complexity of $\mathcal{A}$ is not linear. This contradiction concludes the proof of the theorem. □

Remark. In the above proof, we obtained a threshold $1/2$ for α. Given an arbitrary constant integer c, a threshold $\frac{c}{c+1}$ can be obtained by subdividing cn edges instead of only n edges. Hence, one can show that our upper bound $n\log n + o(n\log n)$ on oracle size permitting wakeup with a linear number of messages in graphs with at most n nodes, is asymptotically optimal.

3. ORACLE SIZE FOR THE BROADCAST

In this section we establish almost tight bounds on the minimum oracle size permitting the broadcast with a linear number of messages. In particular, the following upper bound, together with Theorem 2, shows that an efficient wakeup requires strictly more information about the network than an efficient broadcast.

THEOREM 3. *There exists an oracle of size $O(n)$ permitting the broadcast with a linear number of messages in networks with at most n nodes.*

PROOF. We construct an oracle $\mathcal{O}$ and a broadcast algorithm $\mathcal{A}$ using it, which returns a broadcast scheme $\mathcal{B}$ with linear message complexity. We first describe the oracle $\mathcal{O}$. Let $G = (V, E)$ be any n-node network. Every edge $e = \{u, v\} \in E$ is given the weight

$$w(e) = \min\{\mathrm{port}_u(e), \mathrm{port}_v(e)\}.$$

Let $\#_2(w)$ be the number of bits for encoding a non-negative integer w using standard binary representation, that is $\#_2(w) = 1$ if $w \leq 1$, and $\#_2(w) = \lfloor\log w\rfloor + 1$ if $w > 1$. Call the number $\#_2(w(e))$ the *contribution* of the edge e.

CLAIM 3.1. *There exists a spanning tree T_0 of G, for which $\sum_{e\in E(T_0)} \#_2(w(e)) \leq 4n$.*

We establish the claim by constructing a tree T_0 that yields this contribution. The construction is a variant of Kruskal's minimum-weight spanning tree (MST) algorithm (cf. [6]), similar to the one in [5]. It maintains a collection of trees. Initially, each node of G forms a tree on its own. The construction merges these trees into larger trees until it remains with a single tree giving the solution T_0. More precisely, the construction proceeds in phases. Each phase $k \geq 1$ of the construction consists of four steps. At the beginning of the phase, we identify the collection of "small" trees for the phase: $\mathcal{T}_{\text{small}}(k) = \{T : |T| < 2^k\}$, where $|T|$ denotes the size (number of nodes) of a tree T. Second, for each tree $T \in \mathcal{T}_{\text{small}}(k)$, we look at the set $S(T)$ of edges that connect T to $G \setminus T$, and select an edge $e(T)$ of minimum weight in $S(T)$. (Note that $S(T) \neq \emptyset$ since the graph G is connected.) Third, we add these edges to the collection of trees, thus merging the trees into subgraphs. Each subgraph may contain a cycle, thus, finally, for the last of the four steps, in each subgraph we arbitrarily select one of the edges on the cycle and erase it, effectively transforming the subgraph back into a tree. This process is continued until a single tree remains, which is the desired tree T_0.

To prove the claim, let us denote the collection of trees at the beginning of the kth phase, $k \geq 1$, by $T_1^{(k)}, \ldots, T_{q_k}^{(k)}$, where q_k is the number of trees maintained in phase k. We have $q_1 = n$, and $|T_i^{(1)}| = 1$ for any $1 \leq i \leq n$. Moreover, $\sum_{i=1}^{q_k} |T_i^{(k)}| = n$ for every $k \geq 1$. By induction, we easily get that $|T_i^{(k)}| \geq 2^{k-1}$ for every $k \geq 1$ and $1 \leq i \leq q_k$. Thus $q_k \leq n/2^{k-1}$ for every $k \geq 1$. In particular, the number of phases of the construction is at most $\lceil \log n \rceil$.

Assume that, when considering a small tree $T_i^{(k)}$ in the kth phase, the edge $e(T_i^{(k)})$ incident to some node x of $T_i^{(k)}$ was selected. The only edges incident to node x excluded from consideration are the at most $|T_i^{(k)}| - 1$ edges leading from x to the other nodes in $T_i^{(k)}$. Hence even if all of these edges are "lighter" than the edges leading outside the tree, the port number used for $e(T_i^{(k)})$ is at most $|T_i^{(k)}| - 1$, hence $w(e(T_i^{(k)})) \leq |T_i^{(k)}| - 1$. Therefore

$$\begin{cases} \#_2(w(e(T_i^{(k)}))) = 1 & \text{if } k = 1 \\ \#_2(w(e(T_i^{(k)}))) \leq \lfloor \log(|T_i^{(k)}| - 1) \rfloor + 1 & \text{if } k > 1 \end{cases}$$

For $T_i^{(k)} \in \mathcal{T}_{\text{small}}(k)$, we have $\log |T_i^{(k)}| < k$. Since outgoing edges are selected only for small trees, we have

$$\#_2 \left(w \left(e \left(T_i^{(k)} \right) \right) \right) \leq k\,.$$

Hence the total contribution C_k of the edges added to the structure throughout the kth phase satisfies

$$C_k \leq k\, |\mathcal{T}_{\text{small}}(k)| \leq k\, q_k \leq k\, n/2^{k-1}\,.$$

Therefore, the total contribution $\sum_{k \geq 1} C_k$ of all edges of the resulting tree T_0 satisfies $\sum_{k \geq 1} C_k \leq \sum_{k \geq 1} kn/2^{k-1} \leq 4n$. This completes the proof of Claim 3.1.

For every edge $e = \{u, v\} \in E(T_0)$, the oracle $\mathcal{O}$ assigns the binary representation of $w(e)$ to the extremity $x \in \{u, v\}$ such that $w(e) = \text{port}_x(e)$, where ties are broken arbitrarily. The same node may receive binary representations of several weights $w(e_1), \ldots, w(e_t)$, in which case they can be encoded by one binary string of length $2 \sum_{i=1}^{t} \#_2(w(e_i))$. In view of Claim 3.1, the size of the oracle is at most $8n$.

Based on the strings assigned to the nodes of G by oracle $\mathcal{O}$, Algorithm $\mathcal{A}$ constructs the broadcast scheme $\mathcal{B}$ defined in Figure 1.

```
/* Broadcast Scheme B executed by node x.
     M is the source message. */
begin
    K_x ← list of port numbers given by the oracle O to x;
          /* K_x = incident edges known by x */
    H_x ← K_x; /* H_x = incident edges through which "hello"
          messages may be sent */
    S_x ← ∅; /* S_x = incident edges through which the
          message M has been transmitted */
    repeat
        if x receives the message M via port p then
            K_x ← K_x ∪ {p};
            S_x ← S_x ∪ {p};
        if x receives the message M then
            send message M on all ports of K_x \ S_x;
            S_x ← K_x;
            H_x ← H_x \ S_x;
        if H_x ≠ ∅ then
            send "hello" messages on all ports of H_x;
            H_x ← ∅;
        if x receives a "hello" message via port p ∉ K_x then
            K_x ← K_x ∪ {p};
    endrepeat
end
```

Figure 1: Broadcast Scheme $\mathcal{B}$

CLAIM 3.2. *The scheme $\mathcal{B}$ has linear message complexity, and achieves broadcast in G.*

We establish the first part of the claim by combining the following properties. Clearly, the source message M as well as the "hello" messages are sent only through the $n-1$ edges of T_0. The message M does not traverse an edge more than once because M is sent by x only through edges of $K_x \setminus S_x$, where S_x is the set of edges through which either M has been sent by x before, or M has been received by x. A "hello" message traverses an edge e of T_0 in one direction only because only one extremity x of e is given the port number $\text{port}_x(e)$ by the oracle.

The second part of the claim is established by induction on the distance d of a node from the source, in the tree T_0. Let $P(d)$ be the property "all nodes at distance $\leq d$ from the source in T_0 eventually receive the message M". $P(0)$ clearly holds. Assume $P(d)$ holds for $d \geq 0$, and consider a node x at distance $d+1$ from the source in T_0. Node x is a neighbor in T_0 of a node y at distance d from the source in T_0. The edge $e = \{x, y\}$ is eventually discovered by y because, by definition of the oracle $\mathcal{O}$, either y is given $\text{port}_y(e)$, or x is given $\text{port}_x(e)$, and, in the latter case, x will eventually send a message "hello" to y, enabling e to be known by y. By the induction hypothesis, y will eventually receive the message M. Therefore the message will eventually be sent through e by y. Therefore $P(d+1)$ holds too, and hence $\mathcal{B}$ achieves broadcast. $\square$

THEOREM 4. *Any broadcast algorithm using an oracle of size $o(n)$ in networks with at most n nodes, cannot return a broadcast scheme of linear message complexity.*

PROOF. The proof uses a similar construction as for proving Theorem 2, but requires novel ideas, since the nodes can now transmit spontaneously. Recall that K_n^* denotes the n-node complete graph K_n whose vertices are labeled from 1 to n, and the port number at node i of the edge leading to node j is labeled $(i-j) \bmod (n-1)$. For any k and n such that $4k$ divides n, and for any (n/k)-tuple $S = (e_1, \ldots, e_{n/k})$ of distinct edges of K_n^*, let us consider the graphs obtained from K_n^* by replacing edge e_i by a clique H_i of size k, for $i = 1, \ldots, n/k$. More precisely, one edge $\{a_i, b_i\}$ of the clique H_i replacing e_i is removed from H_i, and a_i is connected to one extremity of e_i in K_n^*, while b_i is connected to the other extremity of e_i in K_n^*. Nodes of H_i are labeled from $n+(i-1)k+1$ to $n+ik$, for $i = 1, \ldots, n/k$. The port number at node $n+(i-1)k+a$ of the edge leading to node $n+(i-1)k+b$ is labeled $(a-b) \bmod (k-1)$. By abuse of notation, the edge $\{n+(i-1)k+a, n+(i-1)k+b\}$ is called the edge $\{a, b\}$ of H_i. The set S does not fully specify the graph resulting from the above transformation because edges $\{a_i, b_i\}$ are not yet specified. Let

$$\mathcal{C} = \Big\{ \Big((a_1, b_1), \ldots, (a_{n/k}, b_{n/k})\Big) \mid (a_i, b_i) \in \{1, \ldots, k\}^2, a_i < b_i, i = 1, \ldots, n/k \Big\}.$$

Any $C \in \mathcal{C}$ (together with the set S) fully characterizes the graph as follows. For any edge e_i in S, $i = 1, \ldots, n/k$, let $e_i = \{u_i, v_i\}$, where $\mathrm{id}(u_i) < \mathrm{id}(v_i)$. The edge e_i of K_n^* and the edge $f_i = \{a_i, b_i\}$ of H_i are replaced by the edges $\{a_i, u_i\}$ and $\{b_i, v_i\}$. The port number at u_i (resp., v_i) of the edge $\{a_i, u_i\}$ (resp., $\{b_i, v_i\}$) is the same as the port number at u_i (resp., v_i) of the edge e_i. Similarly, the port number at a_i (resp., b_i) of the edge $\{a_i, u_i\}$ (resp., $\{b_i, v_i\}$) is the same as the port number at a_i (resp., b_i) of the edge f_i. The resulting graph is denoted by $G_{n,S,C}$. Let the node with label 1 be the source. For any pair of positive integers (n, k) such that $4k$ divides n, the family of graphs defined as above is denoted by $\mathcal{G}_{n,k}$. In other words, we have

$$\mathcal{G}_{n,k} = \{G_{n,S,C} \mid S \text{ is a } (n/k)\text{-tuple of edges of } K_n^*,\ C \in \mathcal{C}\ \}.$$

Note that, by construction, every graph in $\mathcal{G}_{n,k}$ has $2n$ nodes, and in each such graph, all nodes with labels larger than n (i.e., those in the added cliques) have degree $k-1$.

Claim 3.3. *For n and k large enough, such that $k \leq \sqrt{\log n}$ and $4k$ divides n, any broadcast algorithm using an oracle of size at most $\frac{n}{2k}$, for all graphs in $\mathcal{G}_{n,k}$, cannot return a broadcast scheme with message complexity less than $n(k-1)/8$.*

To establish the claim, let us assume, for the purpose of contradiction, that there exists a broadcast algorithm $\mathcal{A}$ using an oracle $\mathcal{O}$ of size at most $\frac{n}{2k}$, for all graphs in $\mathcal{G}_{n,k}$, which produces a broadcast scheme of message complexity less than $n(k-1)/8$. Let σ_i be the sum of numbers of bits given by the oracle $\mathcal{O}$ to the nodes of H_i. Since $\Sigma_{i=1}^{n/k} \sigma_i \leq \frac{n}{2k}$, we get that at least half of the cliques do not receive any bit of information. On the other hand, if

$$\mathcal{A}(\varnothing,\ 0, n+(i-1)k+a,\ k-1)$$

is not defined for some pair (i, a), where $1 \leq i \leq n/k$ and $1 \leq a \leq k$, then at least one node of H_i requires some information from the oracle to specify its broadcast scheme, and thus the clique H_i must receive at least one bit of information. Such an index i is called *heavy*. Let $i \in \{1, \ldots, n/k\}$ be a non heavy index (i.e., i is such that $\mathcal{A}(\varnothing,\ 0, n+(i-1)k+a,\ k-1)$ is defined for all $a = 1, \ldots, k$), and let us observe the behavior of the communication scheme produced by $\mathcal{A}$ in the clique H_i, when the oracle gives no information to the nodes of H_i. If in the synchronous execution of the scheme, all edges of H_i are eventually traversed by at least one message, then i is called *internal*. Otherwise, i.e., if the communication scheme leaves at least one edge of H_i not traversed by any message in the synchronous execution of the scheme, then i is called *external*. External indices result from the fact that the scheme exchanges messages but lets always one edge free of message, or result from the fact that the execution of the scheme reaches a point at which the action of a node is not defined (the history of the execution cannot be produced by the broadcast scheme returned by $\mathcal{A}$).

For every internal index i, let us consider the synchronous execution of the scheme, and let $f_i = \{a_i, b_i\}$ be an edge of H_i that is traversed last. For every external index i, let us again consider the synchronous execution of the scheme, and let f_i be any edge of H_i that is not traversed by any message. Finally, for every heavy index i, let f_i be any edge of H_i. This setting of the f_i's defines one (n/k)-tuple from $\mathcal{C}$, denoted by C^*. We will now restrict attention to those graphs in $\mathcal{G}_{n,k}$, for which S takes all possible values of (n/k)-tuples of edges of K_n^*, but $C = C^*$.

Fix S and consider G_{n,S,C^*}. As observed before, at least half of the cliques in G_{n,S,C^*} receive no information from the oracle. Let I be the corresponding set of indices. We have $|I| \geq n/(2k)$. Indices in I are either internal or external because cliques with heavy indices must receive at least one bit of information from the oracle. Hence I can be decomposed into two sets I_{int} and I_{ext} that are subsets of internal and external indices, respectively, and such that $I = I_{int} \cup I_{ext}$. For all cliques H_i with $i \in I_{ext}$, the setting of the f_i's implies that the broadcast scheme generated by $\mathcal{A}$ has the property that, in its synchronous execution, no message goes out of H_i before a message goes into H_i from the rest of the graph. Among all cliques H_i with $i \in I_{int}$, some may have the property that, in the synchronous execution of the broadcast scheme, a message goes out of H_i before any message goes into H_i from the rest of the graph. Let I_{int}^+ be the indices from I_{int}, for which this phenomenon occurs. By the setting of the f_i's, for every $i \in I_{int}^+$, the message complexity of the broadcast scheme restricted to H_i is at least $k(k-1)/2$ since f_i is one of the edges traversed last. Therefore, since the broadcast scheme generated by $\mathcal{A}$ has message complexity less than $n(k-1)/8$, we get that $|I_{int}^+| < \frac{n}{4k}$. Thus, $|I \setminus I_{int}^+| \geq \frac{n}{4k}$. This inequality implies that the number of cliques H_i such that, in the synchronous execution of the broadcast scheme, no message goes out of H_i before a message goes into H_i from the rest of the graph, is at least $\frac{n}{4k}$.

In other words, at least $\frac{n}{4k}$ cliques have to be discovered from the outside, and at most $\frac{3n}{4k}$ can reveal themselves spontaneously to the rest of the graph. Therefore, the broadcast problem in G_{n,S,C^*} is at least as hard as the auxiliary problem `edge_discovery` with instances (n, X, Y) satisfying $|X| = \frac{n}{4k}$ and $|Y| = \frac{3n}{4k}$. For n, $|X|$, and Y fixed, there are

$$|X|! \binom{\binom{n}{2} - |Y|}{|X|}$$

different instances of `edge_discovery`. Hence, for $|X| = \frac{n}{4k}$ and $|Y| = \frac{3n}{4k}$, the number of different instances $P = |X|!\ P'$ satisfies

$$P' = \binom{\binom{n}{2} - \frac{3n}{4k}}{\frac{n}{4k}} \geq \left(\frac{\binom{n}{2} - \frac{3n}{4k}}{\frac{n}{4k}}\right)^{\frac{n}{4k}} \geq$$

$$\left(\frac{\frac{n^2}{4} - \frac{3n}{4k}}{\frac{n}{4k}}\right)^{\frac{n}{4k}} \geq (nk-3)^{\frac{n}{4k}} \geq \left(\frac{nk}{2}\right)^{\frac{n}{4k}} \quad (7)$$

where the first inequality follows from Equation 1. On the other hand, let Q be the number of possible functions output by an oracle of size at most q for the graphs of $\mathcal{G}_{n,k}$. By the same calculations as for deriving Equation 3, we get

$$Q \leq (q+1)2^q \binom{2n+q}{q}.$$

It follows from Equation 4 that

$$\binom{2n + \frac{n}{2k}}{\frac{n}{2k}} = \binom{\frac{n}{2k}(1+4k)}{\frac{n}{2k}} \leq \big(e(1+4k)\big)^{\frac{n}{2k}} \leq (24k)^{\frac{n}{2k}}$$

for n and k large enough. Since $\frac{n}{2k} + 1 < \frac{n}{k}$, we get

$$Q \leq \frac{n}{k}\, 2^{\frac{n}{2k}}\ (24k)^{\frac{n}{2k}}, \text{ if } q \leq \frac{n}{2k}.$$

Therefore, for n and k large enough,

$$Q \leq (50k)^{\frac{n}{2k}}. \quad (8)$$

There exists a set of graphs of size at least P/Q for which the oracle returns the same function. Combining Equations 7 and 8, we conclude that there exists a set of graphs of size at least

$$|X|!\ \left(\frac{n}{5000\,k}\right)^{\frac{n}{4k}}$$

for which the oracle returns the same function. Applying Lemma 1 to this set of graphs, we get that the number of exchanged messages is at least $\frac{n}{4k}\ \log(\frac{n}{5000\,k})$. For $k \leq \sqrt{\log n}$, and for n and k large enough, this number is at least $n(k-1)/8$, a contradiction with the hypothesis that $\mathcal{A}$ produces a broadcast scheme of message complexity less than $n(k-1)/8$. This completes the proof of Claim 3.3.

To complete the proof of the theorem, let us consider a broadcast algorithm $\mathcal{A}$ using an oracle $\mathcal{O}$ of size $f(n)$ in networks of at most n nodes, where $f(n)$ is in $o(n)$. Let $\widehat{f}$ be the function defined by $\widehat{f}(n) = \max\{f(n), \frac{n}{\sqrt{\log n}}\}$. Hence $\mathcal{A}$ uses an oracle $\mathcal{O}$ of size at most $\widehat{f}(n)$ in networks of at most n nodes. For any $n \geq 1$, let $k(n) = n/\widehat{f}(n)$, and let $k'(n) = \lfloor \frac{k(n)}{4} \rfloor$. Let n' be the largest integer smaller or equal to n and divisible by $4k'(n)$. Note that, since $n/k'(n)$ grows to infinity, we have $n' \geq n/2$, for n large enough. The oracle $\mathcal{O}$ has size at most $\widehat{f}(n)$ in networks with at most n' nodes. We have

$$\widehat{f}(n) = \frac{n}{k(n)} \leq \frac{2n'}{k(n)} \leq \frac{n'}{2\,k'(n)}.$$

Therefore, $\mathcal{O}$ has size at most $\frac{n'}{2\,k'(n)}$ in networks with at most n' nodes. By the construction of $\widehat{f}$, we get $k'(n) \leq \sqrt{\log n'}$. Hence Claim 3.3 applies, and we conclude that the broadcast scheme returned by $\mathcal{A}$ on graphs with at most n' nodes has message complexity at least $n'(k'(n)-1)/8$, which is not in $O(n')$. Therefore, any broadcast algorithm $\mathcal{A}$ using an oracle $\mathcal{O}$ of size $f(n)$ in networks with at most n nodes, where $f(n)$ is in $o(n)$, returns a broadcast scheme that does not have a linear message complexity. □

4. CONCLUSION

We investigated oracles: a new way of modeling knowledge that nodes have about the network. We showed that the minimum oracle size for which a task can be accomplished efficiently, can serve as a measure of difficulty of this task, and can be used to quantitatively differentiate the difficulty of related tasks. In this paper we concentrated on two similar communication tasks, broadcast and wakeup with a linear number of messages, and used oracle size to strictly separate their difficulty. However, we conjecture that oracles can be also used to assess difficulty of a broader range of distributed network problems, not only concerning information dissemination but also, e.g., spanner construction or exploration by mobile entities. Moreover, oracles could be potentially used to establish precise tradeoffs between the amount of knowledge available to nodes of a network and the efficiency (in terms of time or message complexity) of accomplishing a given task.

5. REFERENCES

[1] S. Abiteboul, H. Kaplan, and T. Milo, Compact labeling schemes for ancestor queries. Proc. 12th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA 2001), 547-556.

[2] B. Awerbuch, O. Goldreich, D. Peleg and R. Vainish, A trade-off between information and communication in broadcast protocols, Journal of the ACM 37 (1990), 238-256.

[3] M.A. Bender, A. Fernandez, D. Ron, A. Sahai and S. Vadhan, The power of a pebble: Exploring and mapping directed graphs, Information and Computation 176 (2002), 1-21.

[4] A.E.F. Clementi, A. Monti and R. Silvestri, Selective families, superimposed codes, and broadcasting on unknown radio networks, Proc. 12th Ann. ACM-SIAM Symposium on Discrete Algorithms (SODA 2001), 709-718.

[5] R. Cohen, P. Fraigniaud, D. Ilcinkas, A. Korman and D. Peleg, Labeling Schemes for Tree Representation. Proc. 7th Int. Workshop on Distributed Computing (IWDC 2005), LNCS 3741, 13-24.

[6] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. Introduction to Algorithms. MIT Press, McGraw-Hill, 1990.

[7] A. Czumaj and W. Rytter, Broadcasting algorithms in radio networks with unknown topology, Proc. 44th Ann. Symposium on Foundations of Computer Science (FOCS 2003), 492-501.

[8] A. Dessmark and A. Pelc, Optimal graph exploration without good maps, Theor. Comput. Sci. 326 (2004), 343-362.

[9] F. Fich and E. Ruppert, Hundreds of impossibility results for distributed computing, Distributed Computing, 16 (2003), 121-163.

[10] L. Gasieniec, D. Peleg, and Q. Xin, Faster communication in known topology radio networks,

Proc. 24th Ann. ACM Symposium on Principles of Distributed Computing (PODC 2005), 129-137.

[11] C. Gavoille, D. Peleg, S. Pérennes, and R. Raz, Distance labeling in graphs. Proc. 12th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA 2001), 210-219.

[12] T. Kameda and M. Yamashita, Computing on anonymous networks: Part I – characterizing the solvable cases. IEEE Transactions on Parallel and Distributed Systems, 7 (1996), 69-89.

[13] M. Katz, N. Katz, A. Korman, and D. Peleg, Labeling schemes for flow and connectivity. Proc. 13th Ann. ACM-SIAM Symp. on Discrete algorithms (SODA 2002), 927-936.

[14] E. Korach, S. Moran, and S. Zaks, The Optimality of Distributive Constructions of Minimum Weight and Degree Restricted Spanning Trees in a Complete Network of Processors. SIAM J. Comput. 16(2) (1987), 231-236.

[15] E. Korach, S. Moran, and S. Zaks, Optimal Lower Bounds for Some Distributed Algorithms for a Complete Network of Processors. Theor. Comput. Sci. 64(1) (1989) 125-132.

[16] A. Korman, S. Kutten, and D. Peleg, Proof labeling schemes. Proc. 24th Ann. ACM Symp. on Principles of Distributed Computing (PODC 2005), 9-18.

[17] D. Kowalski and A. Pelc, Optimal deterministic broadcasting in known topology radio networks, manuscript.

[18] F. Thomson Leighton. Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes. Morgan Kaufmann Publishers, 1992.

[19] N. Lynch. A hundred impossibility proofs for distributed computing. Proc. 8th Ann. ACM Symposium on Principles of Distributed Computing (PODC 1989),1-28.

[20] M. Thorup and U. Zwick, Approximate distance oracles. Journal of the ACM 52(1) (2005), 1-24.